Remarks

The present Response is to the Office Action mailed 02/23/2010, made final. Claims 1-28 are standing for examination.

Detailed Action

1. This action is responsive to communications: The Amendment filed 12/01/09.

2. Claims 1-28 are pending in the case. Claims 1, 12, and 18 are independent claims.

Applicant's response:

Acknowledged

Claim Rejections - 35 USC § 103

4. Claims 1-28 are rejected under 35 U.S.C. 103(a) as being unpatentable over DaCosta et al (US-6,826,553 11/30/04) in view of Weinberg et al (US-6,360,332 03/19/02) in further view of Heninger (US-6,029,207 02/22/00).

-In regard to substantially similar independent claims 1 and 12, DaCosta teaches an application for enabling automated notification of applied structural changes to electronic information pages on a network comprising:

an interface for enabling users to build and modify network navigation and interaction templates using a plurality of functional logic blocks for automatically navigating to and interacting with interactive electronic information pages on the network (column 2, lines 11-30 & 55-67; column 3, lines 8-13, 35-43, & 53-65; column 5, lines 30-67; column 7, lines 17-54)(Figs. 1 & 7);

a navigation interface for integrating the software application to a proxy-navigation system for periodic execution of the templates (column 5, lines 19-20: "automatically repeat these steps in a scheduled manner or when requested");

a change notification module for indicating a navigation and interaction routine has failed and for creating a data file associated with the failed routine (column 18, lines 43-67: "it is known the script has failed…and proper notifications sent to individuals or

entities responsible for the operation of the failing script by email…for example"; column 19, lines 1-15); and

sending proper notifications of the failed script to the developer upon failure of the script (column 6, lines 9-13 & 35-41; column 18, lines 53-67; column 19, lines 1-15). DaCosta does not specifically teach storing the data file in a data repository with a point-of-failure indication identifying at least one functional logic block from the plurality of logic blocks, parameters associated with the failed routine, and an identifier of the associated electronic information page subjected to the navigation. Weinberg teaches storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines (column 6, lines 19-22), and upon failure of the routine, creates a data file (column 2, lines 39-40; column 3, lines 29-43; column 6, lines 19-22; column 17, lines 10-52)(Fig. 5F), the data file comprising a point-of-failure indication within the failed routine identifying at least one functional logic block from the plurality of logic blocks of the template that failed (Fig. 5F: column 17, lines 17-21), parameters of the failure (column 17, lines 35-43), an identifier of the associated electronic page (columns 17-18: lines 62-12)(Fig. 5F: "URL: www.mercint.com"), and stores the data file in the data repository sending notification of the action to the developer (column 2, lines 39-40; column 6, lines 15-23). It would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process of the failure along with the an identifier of the associated web page, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67).

DaCosta teaches wherein each template was created by assembling a plurality functional logic blocks which were part of the navigation and interaction templates containing all of the possible navigation and interaction instructions required by the

navigation system-interface module as defined by the a given user/developer (column 2, lines 20-31: "scripts…that locates and extracts data…precisely locating and extracting the select data with a granularity specified by the user" & lines 57-67: "capability for a user to specify…in an automated manor"; column 5, lines 39-55: "learn and store navigation paths…dialogs and forms that need to be filled…login name and password"; column 7, lines 16-28: "captures each user-generated event."; columns 7-8, lines 55-5: "automatically repeatedly query a web site ... upon a single exemplematic query"; column 9, lines 5-44). Neither DaCosta nor Weinberg specifically teach wherein the defined functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29). It would have been obvious to one of ordinary skill in the art at the time of the invention for the functional logic blocks of DaCosta to have been modular parts of the navigation and interaction templates, because Heninger taught that computer software developers realize that modular interacting software components provide the advantages of being more easily designed, generated, tested, installed, and maintained as well leading to better computer products at a minimal cost (column 1, lines 20-67; column 2, lines 1-24). Thus the modular software components of Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script.

-In regard to dependent claims 2, 13, and 19, DaCosta teaches wherein the network could be the Internet (column 2, line 13: "Internet") and wherein the electronic information page was a web page (column 2, line 13: "web site") on the network.

-In regard to dependent claim 3, DaCosta teaches wherein the logic blocks include site logic blocks, automated site-login blocks, and automated site-registration blocks (column 2, lines 55-67; column 5, lines 37-43).

-In regard to dependent claim 4, DaCosta teaches wherein the software application was an Internet based application executing and running on a server (column 16, lines 10-50: "smart servers and smart clients…preferably installed…device of the user…download an installer file"; column 18, lines 33-41: "scripts are stored at a central repository that is accessible through the Internet"; column 25, lines 16-21).

-In regard to dependent claim 5, DaCosta teaches wherein the application was accessible through a network browser (column 2, lines 10-30: "Browser").

-In regard to dependent claim 6, DaCosta teaches wherein the templates are test routines executed for determining success or failure of the routine (column 6, lines 9-13 & 35-41; column 18, lines 54-65).

-In regard to dependent claim 7, DaCosta teaches wherein the templates are executable instruction orders containing logic blocks (column 2, lines 55-67; column 5, lines 37-55; column 6, lines 57-60; column 7, lines 18-54).

-In regard to dependent claim 8, DaCosta teaches wherein the functional logic blocks are modular and self-installable within the templates (column 2, lines 55-67; column 5, lines 37-55; column 6, lines 57-60; column 7, lines 18-54: "stored in an Extensible Markup Language file…programmatically modify the recorded path")(Fig. 2: 60, 70, 80, 90).

-In regard to dependent claim 9, DaCosta teaches wherein the data files are human readable and are accessed by developers for the purpose of affecting updating of the navigation templates (column 7, lines 29-54; column 18, lines 54-67).

-In regard to dependent claim 10, DaCosta teaches wherein the developers access the application via individual computerized workstations (column 18, lines 34-67)(Fig. 7: "User Developer").

-In regard to dependent claim 11, DaCosta teaches wherein the error notification and data file are performed in the event failure or a client's personalized navigation template (column 6, lines 9-13 & 35-41; column 18, lines 34-67).

-In regard to dependent claim 14, DaCosta teaches wherein the software application was an Internet (column 2, line 13: "Internet") based application executing and running on a server (column 18, lines 26-40).

-In regard to dependent claims 15 and 16, DaCosta teaches wherein a single server system hosting both the proxy navigation software and the software application (column 18, lines 26-40).

-In regard to dependent claim 17, DaCosta teaches wherein software application and the proxy navigation software are integrated as a single application enabling both functions of navigation according to navigation templates and notifying and recoding failed instances of navigation (column 18, lines 26-67).

-In regard to independent claim 18, DaCosta teaches a method for receiving automated notification of random structural changes applied to electronic information pages hosted on a network comprising:

-establishing notification of a failed navigation and interaction routine executed for the purpose of navigating to and interacting with an electronic information page (column 6, lines 9- 13 & 35-41; column 18, lines 34-67: "email or pager notification").

-creating an instance of the failed routine associated with the cause of failure (column 18, lines 43-67: "it is known the script has failed…and proper notifications sent

to individuals or entities responsible for the operation of the failing script by email…for example"; column 19, lines 1-15);

-accessing the notification of the of the failed routine for review purposes (column 6, lines 9-13 & 35-41; column 18, lines 34-67: i.e. developer accesses failed script for re-teaching purposes);

-being able to navigate to the electronic information page identified in the recorded instance (column 6, lines 9-13 & 35-41; column 18, lines 34-67: i.e. developer accesses failed script for re-teaching purposes);

-accessing source information associated with the electronic information page identified in the recorded instance (i.e. re-teaching a new navigation and extraction script by accessing the source information).

-creating new logic block according to the source information and according to information contained in the recorded instance (column 6, lines 9-13 & 35-41; column 18, lines 34-67);

installing the new logic block into existing navigation templates that depend on the updated information for successful function (column 6, lines 9-13 & 35-41; column 18, lines 34- 67; column 19, lines 1-15*).*

DaCosta does not specifically teach wherein the instance of the failed navigation routine was stored for future review including parameters associated with the failed routine that included identification of at least one of a plurality of logic blocks used to build each of the navigation template. Weinberg teaches storing the data file (column 2, lines 39-40; column 6, lines 19-22), wherein the application periodically submits test navigation and interaction routines (column 6, lines 19-22), and upon failure of the routine, creates a data file (column 2, lines 39-40; column 3, lines 29-43; column 6, lines 19-22; column 17, lines 10-52)(Fig. 5F), the data file comprising a point-of failure indication within the failed routine and identifying the logic block of the template that failed (Fig. 5F*:* column 17, lines 17-21), parameters of the failure (column 17, lines 35-43), an identifier of the associated electronic page (columns 17-18: lines 62-12)(Fig. 5F: "URL: www.mercint.com"), and stores the data file in the data repository sending

notification of the action to the developer (column 2, lines 39-40; column 6, lines 15-23). It would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process of the failure along with the an identifier of the associated web page, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67).

DaCosta teaches wherein functional logic blocks were part of the navigation and interaction templates containing all of the possible navigation and interaction instructions required by the navigation system-interface module as defined by the a given user/developer (column 2, lines 20-31: "scripts…that locates and extracts data…precisely locating and extracting the select data with a granularity specified by the user" & lines 57-67: "capability for a user to specify…in an automated manor"; column 5, lines 39-55: "learn and store navigation Paths…dialogs and forms that need to be filled…login name and password"; column 7, lines 16-28: "captures each user-generated event."; columns 7-8, lines 55-5: "automatically repeatedly query a web site ... upon a single exemplematic query"; column 9, lines 5-44). Neither DaCosta nor Weinberg specifically teach wherein the functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Heninger teaches building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-29). It would have been obvious to one of ordinary skill in the art at the time of the invention for the functional logic blocks of DaCosta to have been modular parts of the navigation and interaction templates, because Heninger taught that computer software developers realize that modular interacting software components provide the advantages of being more easily designed, generated, tested, installed, and maintained as well leading to better computer products at a minimal cost (column 1, lines 20-67; column 2, lines 1-24). Thus the modular software components of

Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part of the failed navigation and interaction script.

-In regard to dependent claim 20, DaCosta teaches wherein the navigation routine was performed according to a test navigation template (Fig. 2: i.e. according to the navigation and extraction scripts).

-In regard to dependent claim 21, DaCosta teaches wherein the navigation routine was performed according to a client navigation template (Fig. 7: "User").

-In regard to dependent claim 22, DaCosta teaches wherein the recorded instance of the failed routine was created in the form of a data file and stored in a data repository (column 18, lines 54-67).

-In regard to dependent claim 23, DaCosta teaches wherein the recorded instance of the failed navigation routine was accessed by a software developer (column 6, lines 9-13 & 35-41; column 18, lines 54-67).

-In regard to dependent claim 24, DaCosta teaches wherein navigation was performed by the developer utilizing an instance of a browser installed on a computerized workstation (column 2, lines 11-30).

-In regard to dependent claim 25, DaCosta teaches wherein the new logic was in the form of a modular logic block installable to a navigation template (column 6, lines 9-13 & 35-41; column 18, lines 54-67).

-In regard to dependent claim 26, DaCosta teaches wherein the new logic block self-installs to a depended navigation template (column 6, lines 9-13 & 35-41; column 18,

lines 42- 67: "ensure each of the users has a corrected script as soon as possible, i.e., as soon as it is downloaded to the central repository…running the script").

-In regard to dependent claim 27, DaCosta teaches testing the new logic before the implementation (column 19, lines 1-15: "determine whether it is operating correctly").

-In regard to dependent claim 28, DaCosta teaches creating more than one logic block within a navigation template and wherein more than one block could fail (column 6, lines 9-13 & 35-41; column 18, lines 34-67; column 19, lines 1-15).

Applicant's response:

In response to the instant action applicant has cancelled claims 1-17, and has amended claim 18. Claim 18 as amended recites:

18. (Currently amended) A method for receiving notification of structural changes applied to electronic information pages accessed by a proxy network navigation and interaction system and effecting updates to navigation templates based on the change information, comprising steps of:

(a) establishing notification of ~~a~~ failed execution of one of the navigation templates interacting with electronic information pages on a data-packet-network;

(b) recording parameters associated with the cause of failure, including identification of at least one modular logic block involved at the point of failure;

(c) accessing the recorded instance of the failed navigation template for review purposes;

(d) navigating to the electronic information page identified in the recorded instance;

(e) determining information necessary to repair the logic block involved at the point of failure;

(f) creating a new modular logic block according to the information; and

(g) automatically installing the newly created modular logic block into the navigation template that failed, and into all existing navigation templates that depend on the failed logic block.

Claim 18 now recites that once a logic block is repaired, which may be done by a knowledge worker, that logic block is automatically inserted into all navigation templates that depend on the failed logic block. The references do not, either singly or in combination teach this aspect.

Claim 18 is therefore patentable to the applicant, and claims 19-28 are patentable at least as depended from a patentable claim.

From the action:

Response to Arguments

5. Applicant's arguments filed 12/01/09 have been fully considered but they are not persuasive.

-In regard to substantially similar independent claims 1 and 12, Applicant argues that the newly amended features wherein the navigation and interaction templates are created "by assembling a plurality of functional logic blocks" and wherein the data file comprising a point-of- failure indication within the failed routine identifies at least one functional logic block "from the plurality of logic blocks" in the associate template, are not taught or suggested by the applied prior art. The Examiner respectfully disagrees with the Applicant. As shown above in the rejection, the DaCosta reference clearly teaches wherein the navigation and extraction scripts were composed of a plurality of user defined navigation path and extraction steps created by recording user generated events. Based on the plurality of recorded steps the scripts of DaCosta are able to played back and said recorded steps are repeated such that a result could be periodically retrieved. The plurality of recorded steps of said scripts each individually being able to provide specific functional blocks of logic for navigating and interacting across multiple web pages or

web space as well as operating at the HTML or web page level. The Examiner agrees with the Applicant that neither DaCosta nor Weinberg specifically teach wherein the defined functional logic blocks in the defined interaction scripts were modular parts of the interaction scripts. Thus the Heninger reference has been relied upon to teach the notoriously well known programming technique and benefits of building software components in a modular fashion such that each modular component could be constructed, modified, and tested independently (column 1, lines 20-67; column 2, lines 1-24). DaCosta teaches wherein drastic changes could be made to the web space the recorded scripts operate within and thus when utilizing the scripts the scripts could ultimately fail (column 6, lines 9-13 & 35-41; column 18, lines 33-67; column 19, lines 1-15) preventing the plurality of users of the scripts from properly using the scripts. In view of this, one of ordinary skill in the art at the time of the invention would have appreciated that the concept of the modular software components of Heninger would have provided the developers of DaCosta a better way of maintaining, editing, and correcting failed navigation scripts (column 18, lines 34-67) by allowing the developers to fix only the modular part/step of the failed navigation and interaction script without re-teaching the entire script (column 1, lines 20-67; column 2, lines 1-24). In general the Examiner maintains that the selective correcting/editing/updating of modular function code was notoriously well known in the art at the time of the invention and in and of itself is not considered a novel aspect of the claimed invention.

Additionally, Applicant argues that the Examiner has not shown identification of a functional logic block in a failed navigation routine. The Examiner respectfully disagrees. As discussed above in the rejection, the DaCosta reference clearly shows notifying the proper individuals/developers when navigation and interaction scripts, each containing a plurality of functional logic blocks, fail (column 6, lines 9-13 & 35-41; column 18, lines 33-67; column 19, lines 1-15). The Examiner agrees that the DaCosta does not specifically teach a point-of-failure indication identifying at least one functional logic block from the plurality of logic blocks. The Weinberg reference has been relied upon to clearly teach specifically identifying a given step in a test navigation and interaction

routine wherein said step of said routine failed by identifying at least one functional logic block from the plurality of logic blocks of the template that failed (Fig. 5F: column 17, lines 17-21). In view of the drawings, Weinberg clearly shows recording a point-of-failure indication (Fig. 5F: 88 & 89) within the failed routine, indicating that that verification step failed and thus the status of the test as a whole had failed (column 17, lines 50-52). As discussed before, Weinberg teaches wherein results of the test navigation and interaction routines, including the results of the verification steps were stored for viewing (column 2, lines 39-40). Weinberg also teaches wherein displaying the test results in a hierarchical tree ("report tree") can also display the results of the verification steps graphically within the report tree, such as displaying a green check mark or a red "X" symbol to indicate pass/fail status (column 3, lines 29-43; column 17, lines 10-52). Thus the Weinberg reference indicates to the developer via the report tree the point-in-process has failed by displaying a red "X" symbol in the report tree (Fig. 5F: i.e. Red "X" shows that Test Iteration 4 has failed. The Test Status (90) also shows that the current test status is "Failed"). Thus it would have been obvious to one of ordinary skill in the art at the time of the invention to have stored the failed navigation script of DaCosta and for the proper notifications of the failed script to have included a point in process identifying at least one failed functional logic block, because Weinberg teaches that by storing the failed navigation script, a developer can easily display the results of the navigation and quickly determine the location of the failure of the routine (column 3, lines 29-44). This would have made the re-teaching (i.e. correcting) of the navigation script easier for the developer (column 6, lines 9-13 & 35-41; column 18, lines 42-67).

In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e. "replacement or repair of a logic block occurs sometimes automatically, rather than the requirement of re-recording user interactions") are not recited in the rejected claim(s). Although the claims are interpreted in light of the specification, limitations from the specification are not read into the claims. See *In re Van Geuns,* 988 F.2d 1181, 26 USPQ2d 1057 (Fed. Cir. 1993).

Applicant's response:

The examiner states at the end of the "response to arguments" that certain limitations, i.e. automatic replacement of functional logic blocks, that the applicant argued previously were not taught in the references, are now positively claimed.

Summary

As all of the claims, as amended and argued above, have been shown to be patentable over the art presented by the Examiner, applicant respectfully requests reconsideration and the case be passed quickly to issue.

If any fees are due beyond fees paid with this amendment, authorization is made to deduct those fees from deposit account 50-0534. If any time extension is needed beyond any extension requested with this amendment, such extension is hereby requested.

Respectfully Submitted,
Tim Armandpour et al.

By */Donald R. Boys/*
Donald R. Boys
Reg. No. 35,074

Central Coast Patent Agency, Inc.
3 Hangar Way, Suite D
Watsonville, CA 95076
831-768-1755